Villaverde *et al. BMC Systems Biology* (2015) 9:8
DOI 10.1186/s12918-015-0144-4

BMC Systems Biology

SOFTWARE Open Access

BioPreDyn-bench: a suite of benchmark problems for dynamic modelling in systems biology

Alejandro F Villaverde[1], David Henriques[1,2], Kieran Smallbone[3], Sophia Bongard[4], Joachim Schmid[4], Damjan Cicin-Sain[5,6], Anton Crombach[5,6], Julio Saez-Rodriguez[2], Klaus Mauch[4], Eva Balsa-Canto[1], Pedro Mendes[3], Johannes Jaeger[5,6] and Julio R Banga[1*]

Abstract

Background: Dynamic modelling is one of the cornerstones of systems biology. Many research efforts are currently being invested in the development and exploitation of large-scale kinetic models. The associated problems of parameter estimation (model calibration) and optimal experimental design are particularly challenging. The community has already developed many methods and software packages which aim to facilitate these tasks. However, there is a lack of suitable benchmark problems which allow a fair and systematic evaluation and comparison of these contributions.

Results: Here we present BioPreDyn-bench, a set of challenging parameter estimation problems which aspire to serve as reference test cases in this area. This set comprises six problems including medium and large-scale kinetic models of the bacterium *E. coli*, baker's yeast *S. cerevisiae*, the vinegar fly *D. melanogaster*, Chinese Hamster Ovary cells, and a generic signal transduction network. The level of description includes metabolism, transcription, signal transduction, and development. For each problem we provide (i) a basic description and formulation, (ii) implementations ready-to-run in several formats, (iii) computational results obtained with specific solvers, (iv) a basic analysis and interpretation.

Conclusions: This suite of benchmark problems can be readily used to evaluate and compare parameter estimation methods. Further, it can also be used to build test problems for sensitivity and identifiability analysis, model reduction and optimal experimental design methods. The suite, including codes and documentation, can be freely downloaded from the BioPreDyn-bench website, https://sites.google.com/site/biopredynbenchmarks/.

Keywords: Dynamic modelling, Model calibration, Parameter estimation, Optimization, Benchmarks, Large-scale, Metabolism, Transcription, Signal transduction, development

Background

Systems biology aims at understanding the organization of complex biological systems with a combination of mathematical modelling, experiments, and advanced computational tools. To describe the behaviour of complex systems, models with sufficient level of detail to provide mechanistic explanations are needed. This leads to the use of large-scale dynamic models of cellular processes [1]. By incorporating kinetic information, the range of applications of biological models can be widened. The importance of kinetic models is being increasingly acknowledged in fields such as bioprocess optimization [2], metabolic engineering [3], physiology, as well as cell and developmental biology [4].

Systems identification, or reverse engineering, plays an important part in the model building process. The difficult nature of reverse engineering was stressed in [5], where the different perspectives that coexist in the area of systems biology were reviewed. Specifically, large-scale dynamic biological models generally have many unknown,

*Correspondence: julio@iim.csic.es
[1] Bioprocess Engineering Group, IIM-CSIC, Eduardo Cabello 6, 36208 Vigo, Spain
Full list of author information is available at the end of the article

non-measurable parameters. For the models to encapsulate as accurately as possible our understanding of the system (i.e. reproducing the available data and, ideally, being capable of making predictions), these parameters have to be estimated. This task, known as parameter estimation, model calibration, or data fitting [6-10], consists of finding the parameter values that give the best fit between the model output and a set of experimental data. This is carried out by optimizing a cost function that measures the goodness of this fit. In systems biology models this problem is often multimodal (nonconvex), due to the nonlinear and constrained nature of the system dynamics. Hence, standard local methods usually fail to obtain the global optimum. As an alternative, one may choose a multistart strategy, where a local method is used repeatedly, starting from a number of different initial guesses for the parameters. However, this approach is usually not efficient for realistic applications, and global optimization techniques need to be used instead [11,12].

Many methods have been presented for this task, but less effort has been devoted to their critical evaluation. It is clear, however, that to make progress in this research area it is essential to assess performance of the different algorithms quantitatively, in order to understand their weaknesses and strengths. Furthermore, if a new algorithm is to be accepted as a valuable addition to the state of the art, it must be first rigorously compared with the existing plethora of methods. This systematic comparison requires adequate benchmark problems, that is, reference calibration case studies of realistic size and nature that can be easily used by the community. Several collections of benchmarks – and of methods for generating them – have already been published [13-19]. An artificial gene network generator, which allows to choose from different topologies, was presented in [13]. The system, known as A-BIOCHEM, generates pseudo-experimental noisy data in silico, simulating microarray experiments. An artificial gene network with ten genes generated in this way was later used to compare four reverse-engineering methods [15]. More recently, a toolkit called GRENDEL was presented with the same purpose [17], including several refinements in order to increase the biological realism of the benchmark. A reverse-engineering benchmark of a small biochemical network was presented in [14]. The model describes organism growth in a bioreactor and the focus was placed on model discrimination using measurements of some intracellular components. A proposal for minimum requirements of problem specifications, along with a collection of 44 small benchmarks for ODE model identification of cellular systems, was presented in [16]. The collection includes parameter estimation problems as well as combined parameter and structure inference problems. Another method for generation of dynamical models of gene regulatory networks to be used as benchmarks is GeneNetWeaver [19], which was used to provide the international Dialogue for Reverse Engineering Assessments and Methods (DREAM) competition with three network inference challenges (DREAM3, DREAM4 and DREAM5) [18]. Subsequent competitions (DREAM6, DREAM7) included also parameter estimation challenges of medium-scale models [20]. Similar efforts have been carried out in related areas, such as in optimization, where BBOB workshops (Black-Box Optimization Benchmarking, [21]) have been organised since 2009. In this context it is also worth mentioning the collection of large-scale, nonlinearly constrained optimization problems from the physical sciences and engineering (COPS) [22].

Despite these contributions, there is still a lack of suitable benchmark problems in systems biology that are at the same time (i) dynamic, (ii) large-scale, (iii) ready-to-run, and (iv) available in several common formats. None of the above mentioned collections possesses all these features, although each one has a subset of them. Here we present a collection of medium and large-scale dynamic systems, with sizes of tens to hundreds of variables and hundreds to thousands of estimated parameters, which can be used as benchmarks for reverse-engineering techniques. The collection includes two *Escherichia coli* models [23,24], a genome-wide kinetic model of *Saccharomyces cerevisiae* [25], a metabolic model of Chinese Hamster Ovary (CHO) cells [26], a signal transduction model of human cells [27], and a developmental gene regulatory network of *Drosophila melanogaster* [28-30].

Ensuring standardisation allows systems biology models to be reused outside of their original context: in different simulators, under different conditions, or as parts of more complex models [31]. To this end, we have made five of the six models (the exception is the spatial model of *D. melanogaster*) available in Systems Biology Markup Language (SBML [32]) format, allowing for their simulation in multiple software tools, including AMIGO [33] and COPASI [34]. Even when defined in a standard format such as SBML, large models such as the genome-wide kinetic model of *S. cerevisiae* may give different results when simulated in different software environments. The inherent size and stiffness of genome-scale systems biology models create new challenges to be addressed for their robust simulation by systems biology tools [25]. To address this problem all the models have been consistently formatted, with their dynamics provided both in C and in Matlab. Additionally, a benchmark consisting of a parameter estimation problem has been defined for every model, for which ready-to-run implementations are provided in Matlab (optionally with the use of the AMIGO toolbox [33]) and, in some cases, also in COPASI [34]. The availability of ready-to-run implementations is a highly desirable practice in computer science, since it ensures

reproducibility of the results. Calibration results with state of the art optimization methods are reported, which can serve as a reference for comparison with new methodologies. Additionally, suggestions on how to compare the performance of several methods are also given in the Results and discussion section.

Problem statement

Given a model of a nonlinear dynamic system and a set of experimental data, the parameter estimation problem consists of finding the optimal vector of decision variables p (unknown model parameters). This vector consists of the set of parameter values that minimize a cost function that measures the goodness of the fit of the model predictions with respect to the data, subject to a number of constraints. The output state variables that are measured experimentally are called observables. The following elements need to be clearly stated in order to properly define the calibration problem:

- cost function to optimize (i.e. metrics which reflects the mismatch between experimental and predicted values)
- dynamics of the systems (in our benchmark models they are given by systems of ordinary differential equations)
- model parameters to be estimated
- initial conditions for the dynamics (possibly unknown, in which case they are included among the parameters to be estimated)
- upper and lower bounds for the parameters
- state variables that can be measured (observed)
- values of external stimuli, also known as control variables
- measurements (over time and/or space) available for the calibration: number of experiments, stimuli (if any) for each experiment, data points per experiment, etc.
- (optional) type and magnitude of errors considered for the experimental data
- (optional) additional time series for validation of the calibrated model
- solver used to numerically simulate the systems, and the relative and absolute error tolerances used

Mathematically, it is formulated as a nonlinear programming problem (NLP) with differential-algebraic constraints (DAEs), where the goal is to find p to minimize an objective function. The objective function, or cost function, is a scalar measure of the distance between data and model predictions. There are several common choices for the objective function. The generalized least squares cost function is given by:

$$J_{lsq} = \sum_{\epsilon=1}^{n_\epsilon} \sum_{o=1}^{n_o^\epsilon} \sum_{s=1}^{n_s^{\epsilon,o}} w_s^{\epsilon,o} \left(ym_s^{\epsilon,o} - y_s^{\epsilon,o}(p) \right)^2 \tag{1}$$

where n_ϵ is the number of experiments, n_o^ϵ is the number of observables per experiment, and $n_s^{\epsilon,o}$ is the number of samples per observable per experiment. The measured data will be denoted as $ym_s^{\epsilon,o}$ and the corresponding model predictions will be denoted as $y_s^{\epsilon,o}(p)$. Finally, $w_s^{\epsilon,o}$ are scaling factors used to balance the contributions of the observables, according to their magnitudes and/or the confidence in the measurements. When information about the experimental error is available, one may use the maximum (log-)likelihood function to look for the parameters with the highest probability of generating the measured data. Assuming independently identically distributed measurements with normally distributed noise, the likelihood is defined as:

$$J_{llk} = \sum_{\epsilon=1}^{n_\epsilon} \sum_{o=1}^{n_o^\epsilon} \sum_{s=1}^{n_s^{\epsilon,o}} \frac{\left(ym_s^{\epsilon,o} - y_s^{\epsilon,o}(p) \right)^2}{(\sigma_s^{\epsilon,o})^2} \tag{2}$$

For known constant variances the log-likelihood cost function is similar to the generalized least squares, with weights chosen as the inverse of the variance, $w_s^{\epsilon,o} = 1/\left(\sigma_s^{\epsilon,o}\right)^2$. This is the case of most of the benchmark problems presented here (B1, B2, B4, B5). The exceptions are: problem B3, in which no noise has been added to the data, and thus the scaling factors $w_s^{\epsilon,o}$ are taken as the squared inverse of the maximum experimental value for each observable; and problem B6, where the weights are inversely related to the level of expression. More details are given in the next section. Note that the cost functions used in Matlab/AMIGO are not exactly the same as the ones used in COPASI, since in COPASI the weights are scaled so that for each experiment the maximal occurring weight is 1.

The minimization of the objective function is subject to the following constraints:

$$\dot{x} = f(x, p, t) \tag{3}$$

$$x(t_0) = x_0 \tag{4}$$

$$y = g(x, p, t) \tag{5}$$

$$h_{eq}(x, y, p) = 0 \tag{6}$$

$$h_{in}(x, y, p) \leq 0 \tag{7}$$

$$p^L \leq p \leq p^U \tag{8}$$

where g is the observation function, x is the vector of state variables with initial conditions x_0, f is the set of differential and algebraic equality constraints describing the

system dynamics (that is, the nonlinear process model), h_{eq} and h_{in} are equality and inequality constraints that express additional requirements for the system performance, and p^L and p^U are lower and upper bounds for the parameter vector p. The problem defined above is the general formulation of a nonlinear least squares optimization subject to dynamic constraints and bounds in the parameters. The problems included in this collection of benchmarks do not make use of constraints (6–7).

Remarks on parameter estimation methods

Fitting a large, nonlinear model to experimental (noisy) data is generally a multimodal problem. In these circumstances, the use of local optimization methods, which are usually gradient-based, entails the risk of converging to local minima. Hence it is needed to use global optimization methods that provide more guarantees of converging to the globally optimal solution [11,35]. Global optimization strategies can be roughly classified as deterministic, stochastic and hybrid. Deterministic methods can guarantee the location of the global optimum solution; however, their computational cost makes them unfeasible for large-scale problems. Stochastic methods, which are based on probabilistic algorithms, do not provide those guarantees, but are frequently capable of finding optimal or near-optimal solutions in affordable computation times.

Some of the most efficient stochastic global optimization methods are the metaheuristic approaches. A heuristic is an algorithm originated not from formal analysis, but from an expert knowledge of the task to be solved. A metaheuristic can be seen as a general-purpose heuristic method designed to guide an underlying problem-specific heuristic. It is therefore a method that can be applied to different optimization problems with few modifications. Hybrid methods which combine metaheuristics for global optimization and local methods for accelerating convergence in the vicinity of local minima can be particularly efficient. One such method is the enhanced Scatter Search algorithm, eSS [36], and its parallel cooperative version, CeSS [37]. Matlab and R implementations are publicly available as part of the MEIGO toolbox [38]. The eSS method is available as a Matlab toolbox and is also included in AMIGO; this latter version is the one used in this work. It should be noted that AMIGO offers more than a dozen optimization solvers, including local and global methods, and the possibility of combining them to form user-defined sequential hybrid methods. In COPASI [34] it is possible to choose among thirteen different optimization methods for parameter estimation, including deterministic and stochastic: Evolutionary Programming, Evolutionary Strategy (SRES), Genetic Algorithm, Hooke and Jeeves, Levenberg–Marquardt, Nelder–Mead, Particle Swarm, Praxis, Random Search, Simulated Annealing, Scatter Search, Steepest Descent, and Truncated Newton.

Remarks on comparing optimization methods

Although the objective of this paper is to present a set of ready-to-run benchmarks, we list below several guidelines on how to compare different optimizers with these problems.

Many optimization methods require an initial point and/or bounds on the decision variables. For ensuring a fair comparison between different methods, the same bounds and initial points should be set. Obviously, the nominal solution can not be used as an initial point (note that in this work we use the term "nominal" to refer to the "true" or "reference" parameter values, i.e. for problems that use pseudo-experimental data the nominal solution is the parameter vector that was used to generate the data). Special emphasis should be laid on ensuring full reproducibility. This entails providing all source codes and binary files used in computations, as well as specifying all implementation details, such as software and hardware environment (including compiler versions and options, if any). If some aspects of a method can be tuned, these settings must be clearly indicated.

Many different criteria may be used for comparing the performance of optimization methods. It can be expressed as a function of CPU time, number of function evaluations, or iteration counts. When considering several problems, a solver's average or cumulative performance metric can be chosen. If an algorithm fails to converge a penalty can be used, in which case an additional decision is required to fix its value. An alternative is to use ranks instead of numerical values, although this option hides the magnitudes of the performance metric. All approaches have advantages and drawbacks, and their use requires making choices that are subjective to some extent. In an attempt to combine the best features of different criteria, Dolan and Moré [39] proposed to compare algorithms based on their performance profiles, which are cumulative distribution functions for a performance metric. Performance profiles basically rely on calculations of the ratio of the solver resource time versus the best time of all the solvers. It should be noted that, for complex large-scale problems where identifiability is an issue, different methods often arrive at different solutions. In that case the use of performance profiles requires choosing a tolerance to define acceptable solutions. Performance profiles are a convenient way of summarizing results when there are many methods to be compared and many problems on which to test them. When this is not the case, however, more information can be provided by using convergence curves. Convergence curves plot the evolution of the objective function, as defined in Equation (1), as a function of the number of evaluations or the computation time (since the overhead is different for each method). They provide information not only about the final value reached

by an algorithm, but also about the speed of progression towards that value.

When comparing different optimization methods, the best result (cost) and the mean (or median) for N runs should be reported in a table. Similar statistics for computation and number of evaluations should apply to all the methods. However, since the final values can be greatly misleading, convergence curves should be provided in addition to this table.

Note that, in order to make a fair comparison of convergence curves obtained with different software tools and/or hardware environments, it is a good practice to report any speedup due to parallelism. This can happen in non-obvious situations. For example, COPASI can make use of several threads in multi-core PCs due to its use of the Intel MKL library. In summary, fair comparisons should be made taking into account the real overall computational effort used by each method/implementation. As a general rule, only methods running in the same platform should be compared.

Remarks on identifiability

Parameter estimation is just one aspect of what is known as the inverse problem. This larger problem also includes identifiability analysis, which determines whether the unknown parameter values can be uniquely estimated [40,41]. Lack of identifiability means that there are several possible parameter vectors that give the same agreement between experimental data and model predictions. We may distinguish between a priori structural identifiability and a posteriori or practical identifiability [40,41]. The parameters are structurally identifiable if they can be uniquely estimated from the designed experiment under ideal conditions of noise-free observations and error-free model structure. Structural identifiability is a theoretical property of the model structure, which can be very difficult to determine for large and complex models [42-44]. Even if a model is structurally identifiable, it may exhibit practical identifiability issues. Practical identifiability depends on the output sensitivity functions (partial derivatives of the measured states with respect to the parameters). If the sensitivity functions are linearly dependent the model is not identifiable, and sensitivity functions that are nearly linearly dependent result in parameter estimates that are highly correlated. Furthermore, even if they are linearly independent, low sensitivities may lead to an undesirable situation. Practical identifiability can be studied from sensitivity-based criteria like the Fisher information matrix (FIM). The practical identifiability of the models can be analyzed in this way with the AMIGO toolbox [33]. The AMIGO_LRank method ranks the model parameters according to their influence on the model outputs, using several sensitivity measures. In large biological models identifiability issues are the norm rather than the exception [9,42,45-54]. This may be partly due to inconsistent modelling practices, but even when a model has been carefully built and is structurally identifiable, the amount of data required for a perfect calibration (practical identifiability) is usually large. As an illustration, consider the general case of a model described by differential equations, and assume the ideal situation where the structure of the equations is perfectly known. Then, a well-known result states that identification of r parameter values requires $2r + 1$ measurements [55]. However, it is frequently the case that a model with more than a thousand parameters has to be calibrated with only dozens or maybe hundreds of measurements. Another issue which is intimately related to lack of identifiability is overfitting, which may occur when the number of parameters is too large compared to the number of data. In this case the calibrated model is actually describing the noise present in the data, instead of the true dynamics of the system. Like unidentifiability, overfitting is common in large-scale systems biology models such as the ones presented here. Finally, it should be noted that lack of identifiability does not preclude the use of model-based methods. Unique model predictions can in fact be obtained despite unidentifiability, as discussed by Cedersund [52].

Implementation

Here we present a collection of parameter estimation problems and their descriptions. The characteristics of the six dynamic models are summarized in Table 1. Four of the benchmark problems have been defined using in silico experiments, where pseudoexperimental data have been generated from simulations of the models and addition of artificial noise. The use of simulated data is usually considered the best way of assessing performance of parameter estimation methods, because the true solution is known. Additionally, we provide two benchmark problems that use real data. For each problem we provide the following information (see Additional file 1 and the BioPreDyn-bench website, https://sites.google.com/site/biopredynbenchmarks/):

- Dynamic model.
- Experimental information: initial conditions, input functions, what is measured, measurement times, noise level and type.
- Cost function to be used: its type (least squares, weighted least squares, maximum likelihood, etc), and why it should be chosen.
- Parameters to estimate: lower and upper bounds, initial guesses, nominal values (the latter are reference values, which must not be used during estimations).
- Implementations (Matlab with and without the AMIGO toolbox, C, COPASI): installation, requirements, and usage. Ready-to-run scripts are

Table 1 Models

Model ID	B1	B2	B3	B4	B5	B6
Model Ref	[25]	[23]	[24]	[26]	[27]	[29]
Cell	*S. cerevisiae*	*E. coli*	*E. coli*	CHO	Generic	*Drosophila melanogaster*
Description level	Metabolic: genome scale	Metabolic: CCM	Metabolic: CCM & transcription	Metabolic	Signal transduction	Developmental GRN (spatial)
Parameters	1759	116	178	117	86	37
Dynamic states	276	18	47	34	26	108 –212
Observed states	44	9	47	13	6	108 –212
Experiments	1	1	1	1	10	1
Data points	5280	110	7567	169	96	1804
Data type	simulated	measured	simulated	simulated	simulated	measured
Noise level	$\sigma = 5\%$	real	no noise	variable	$\sigma = 5\%$	real

Main features of the benchmark models. The standard deviation, σ, is given in those cases where artificial noise with constant variance was added to the data.

provided, with examples of how to execute them and their expected output.

Problem B1: genome-wide kinetic model of *S. cerevisiae*

Implementations of the benchmark problems are provided as Additional files 2 and 3. The version of the AMIGO toolbox used in this work is provided as Additional files 4 and 5. The biochemical structure of this model is taken from yeast.sf.net (version 6, [56]). In decompartmentalised form, this network has 1156 reactions and 762 variables. We fix some experimentally determined exchange fluxes, and use geometric FBA [57] to choose a unique reference flux distribution consistent with the experimental data. We fix some initial concentrations to their experimentally determined levels and assign the remainder typical values. We define reaction kinetics using the common modular rate law, a generalised form of the reversible Michaelis-Menten kinetics that can be applied to any reaction stoichiometry [58]. The final model contains 261 reactions with 262 variables and 1759 parameters. This model has been created according to the pipeline presented in [25], which ensures consistency with our sparse data set; whilst no data is required to produce the model, it can incorporate any known flux or concentration data or any kinetic constants. As an addition to the model developed in [25], this version has been alligned with previously unpublished experimental data. The new data consist of 44 steady-state measurements (38 concentrations and 6 fluxes), which are included in Additional file 1: Table S12 and S13. The steady state is found to be stable. The number of measurements available at the present stage is not enough for carrying out a proper model calibration. Envisioning that dynamic (time-series) measurements of the 44 observed variables may be available in the near future, we show in this paper how they will be employed for re-estimating the parameter values. With this aim, we have generated pseudo-experimental noisy data corresponding to a pulse in the concentration of extracellular glucose, and have used this simulated data to re-calibrate the model. We generated 120 samples per observable and added artificial measurement noise (standard uniform distribution, $\sigma = 5\%$) to resemble realistic conditions.

Problem B2: dynamic model of the Central Carbon Metabolism of *E. coli*

This model, originally published in [23] and available at the BioModels database [59], reproduces the response to a pulse in extracellular glucose concentration. It includes 18 metabolites in two different compartments: the cytosol (17 internal metabolites), and the extracellular compartment (1 extracellular metabolite: glucose). These metabolites are involved in 48 reactions: 30 kinetic rate reactions, 9 degradation equations, 8 dilution equations, and 1 equation for extracellular glucose kinetics. Additionally, there are 7 analytical functions, thus the model is defined by a total of 55 mathematical expressions. We have reformulated the model to use it as a parameter estimation problem; the 116 parameters to be estimated consist of kinetic parameters and maximum reaction rates.

As an addition to the model version available in the Biomodels database, we provide the experimental data that were used in the original publication but had not been published (Klaus Mauch, personal communication). The dataset is given in Additional file 1: Table S14, and consists of time-course concentration measurements of nine metabolites. The aim of the model calibration in this case is to find a better fit to the experimental data than the one obtained with the nominal parameter vector used in the original publication [23]. Note that this is different from benchmarks 1 and 3–5, which use simulated data and where the aim is to recover a fit as good as the one

obtained with the nominal parameter vector, with which the data were generated.

Problem B3: enzymatic and transcriptional regulation of the Central Carbon Metabolism of *E. coli*

This model simulates the adaptation of *E. coli* to changing carbon sources. Complete information about this model is available as the supplementary information of [24]. It is also included in the BioModels Database [59]. It should be noted that there are some differences in parameter values between the original model and the BioModels version; however, these changes do not alter the simulation results, a fact that indicates unidentifiability. The model contains 47 ODEs and 193 parameters, of which 178 are considered unknown and need to be estimated. The other 15 parameters are constants known to the modeler (number of subunits of the multimers–enzymes–, scaling factors, universal protein degradation rate, and gene expression rate constant). The outputs of the system are the 47 state variables, which represent concentrations. Pseudo-experimental data were generated by simulation of the sixth scenario defined in the simulation files included as supplementary material in [24]. This scenario simulates an extended diauxic shift which consists of three consecutive environments, where the carbon sources are first glucose, then acetate, and finally a mixture of both. Under these conditions, the 47 concentration profiles are sampled every 1000 seconds, for a total of 161 time points (45 hours). This model exhibits large differences in value among concentrations, which span five orders of magnitude. To equalize their contribution to the objective function, we scale each time-series dividing it by the maximum of the experimental value (scaled least squares).

Problem B4: metabolic model of Chinese Hamster Ovary (CHO) cells

Chinese Hamster Ovary cells (CHO) are used for protein production in fermentation processes [60]. This model simulates a batch process with resting cells: no metabolites are fed for a final time horizon of 300 hours. The fermenter medium contains glucose as main carbon source, and leucine and methionine are the main amino acids taken up. Lactate was modelled to be a by-product of the fermentation process. A generated protein serves as main product of the fermentation process. The model comprises 35 metabolites in three compartments (fermenter, cytosol, and mitochondria) and 32 reactions, including protein product formation, Embden-Meyerhof-Parnas pathway (EMP), TCA cycle, a reduced amino acid metabolism, lactate production, and the electron transport chain. The kinetics are modelled as in [23], and the resulting ODE model comprises 117 parameters in total. Some aspects of this model were partially discussed in [26]. For optimization purposes pseudo-experimental data were generated, mimicking a typical cell behavior. The following 13 metabolites are assumed to be measured: in fermenter, glucose, lactate, product protein, leucine, and methionine; in cytosol, aspartate, malate, pyruvate, oxaloacetate, ATP, and ADP; and in mitochondria, ATP and ADP. Samples were assumed to be daily taken over the whole fermentation time.

Problem B5: signal transduction logic model

To illustrate the advantages and disadvantages of different formalisms related to logic models, MacNamara and colleagues constructed a plausible network of interactions consisting of signaling factors known to be activated downstream of *EGF* and *TNF-α* [27]. The model consists of 26 ODEs that use a logic-based formalism, which is explained in detail in [61]. In this formalism, state values can vary between 0 and 1 and represent the normalized activity of a given protein, which is typically measured as the level of phosphorylation. In total the model includes 86 continuous parameters, corresponding to the half maximal activations (k), the Hill coefficients (n) and a set of parameters controlling the rate of activation/deactivation of a given protein (τ). The model incorporates *EGF* and *TNF-α* which are treated as stimuli that trigger the pathway response. In addition to these two stimuli, the model includes two kinase inhibitors for *RAF*1 and *PI3K*, which can block the activity of both species. In total the model can be perturbed by these 4 cues, allowing a rich variation in the dynamic profiles of the model signaling components, an essential requirement for parameter estimation. In order to generate a data-set for reverse engineering the model structure, the authors generated data, corresponding to 10 in-silico experiments, where the different cues (stimuli and inhibitors) are added in different combinations. For each experiment 6 strategically located states are observed. Each observable was measured at 16 equidistant time points per experiment. In addition to this Gaussian noise was added to the data in order to mimic a reasonable amount of experimental error. Note that the SBML implementation of this model uses the SBML qual format [62], an extension of SBML developed for qualitative models of biological networks.

Problem B6: the gap gene network of the vinegar fly, *Drosophila melanogaster*

Our last benchmark model is slightly different from those previously described, in that it represents a spatial model of pattern formation in multi-cellular animal development, and the data for fitting are based on microscopy, rather than metabolomics or transcriptomics. The gap genes form part of the segmentation gene network, which patterns the anterior–posterior (AP) axis of the *Drosophila melanogaster* embryo. They are the primary regulatory targets of maternal morphogen gradients, and

are active during the blastoderm stage in early development. In the model, the embryo is a single row of dividing nuclei along the AP axis, with each nucleus containing the four gap genes and receiving input from four external factors. The gap genes included in the model are *hunchback* (*hb*), *Krüppel* (*Kr*), *giant* (*gt*), and *knirps* (*kni*), and the external inputs Bicoid (Bcd), Caudal (Cad), Tailless (Tll), and Huckebein (Hkb). Three processes occur within and between nuclei: (1) regulated gene product synthesis, (2) Fickian gene product diffusion, and (3) linear gene product decay. These processes are formalised with ODEs, and result in the model having 37 unknown parameters. This model [29,30] implements the gene circuit approach [63,64] used to reverse-engineer the regulatory interactions of the gap genes by fitting to quantitative spatio-temporal gene expression data, which can be mRNA [29] or protein [30]. The data consist of 9 time points spanning 71 minutes of *Drosophila* development, and at each time point maximally 53 nuclei with data points for the four gap genes, and the four external inputs. The fit is measured with a weighted least squares scheme (WLS) with variable weights, which, in the case of the mRNA data used here [29], are inversely related to the level of expression. The weights were created from normalized, integrated mRNA expression data according to the formula: $w = 1.0 - 0.9y$, with $y \in [0, 1]$ being the normalized staining intensity. This proportionality of variation with expression level reflects the fact that gap domains (showing high levels of expression) show more variation than those regions of the embryo in which a gene is not expressed [29].

Results and discussion

We show how our collection of benchmark problems can be used by reporting selected results using several parameter estimation methods. We emphasize that the purpose of this work is not to provide a comprehensive comparison of all existing approaches, but to provide a useful, versatile, and practical test set and illustrate its use. For simplicity, and to enable direct comparisons among benchmarks, all the computations reported in this section have been carried out in Matlab, using the algorithms available in the AMIGO toolbox [33]. This includes both global and local optimization methods; the latter have been used in a multistart procedure, where multiple instances are launched from different initial points selected randomly within the parameter bounds.

Before estimating the parameter values we assessed the identifiability of the models. Model parameters were ranked according to their influence on the system output (sensitivity), using the local rank routine (AMIGO_LRank) from the AMIGO toolbox as described in the previous section. As is typical of models of this size, it was found that all benchmarks have identifiability issues, with a portion of their parameters exerting very little influence on the model outputs. Therefore, the goal of these benchmarks is not to obtain accurate estimates of all the parameters, but rather to obtain a good fit to the data: when tested on this collection of benchmarks, optimization methods should be evaluated by their ability to minimize the objective function. As an illustration of the typical outcome that can be obtained from the local rank method, we show in Figure 1 the results of the practical identifiability analysis for problem B2. Figure 1 ranks the parameters in decreasing order of their influence on the system's behaviour, which is quantified by means of the importance factors δ_p^{msqr}:

$$\delta_p^{msqr} = \frac{1}{n_{lhs} n_d} \sqrt{\sum_{mc=1}^{n_{lhs}} \sum_{d=1}^{n_d} ([s_d]_{mc})^2} \tag{9}$$

where n_{lhs} are the different values for each of the parameters selected by Latin Hypercube Sampling and n_d is the number of experiments. The relative sensitivities, $[s_d]_{mc}$, measure the influence of every parameter p_{mc} on the model output for a given experiment d. They are defined as

$$[s_d]_{mc} = \frac{1}{n_o \cdot n_s} \sum_{o=1}^{n_o} \sum_{s=1}^{n_s} \frac{\Delta p_{mc}}{\Delta y_d^o} \frac{\delta y_d^o}{\delta p_{mc}} \left(t_s^{d,o}\right) \tag{10}$$

where n_o is the number of observables and n_s the number of sampling times. Figure 1 shows the sensitivity of the state variables with respect to the parameters. From this figure it becomes clear that many parameters such as 8-10, 32-38, 56-64, are not influencing observables. Therefore those parameters are expected to be poorly identifiable.

In the remainder of this section we show selected results of the best performing optimization methods in every parameter estimation problem. Complete results for every benchmark are reported in the Additional file 1.

To evaluate the performance of local methods we launched repeated local searches in a multistart procedure, starting from initial parameter vectors with values chosen randomly from within the parameter bounds. It should be noted that, while multistarts of local searches are a popular option for parameter estimation, they are usually not the most efficient solution when dealing with large-scale nonlinear models. Due to the multimodal nature of these problems, local methods tend to be stuck in local minima, which can sometimes be very far from the global optimum. Launching local methods from random points leads to spending a large fraction of the computational time in unsuccessful searches. Hence, global optimization methods usually perform better in these cases, especially if–as happens with eSS–they are used in combination with local searches. As an example, Figure 2 shows histograms of the results (i.e., objective function values reached and the frequency with which they were

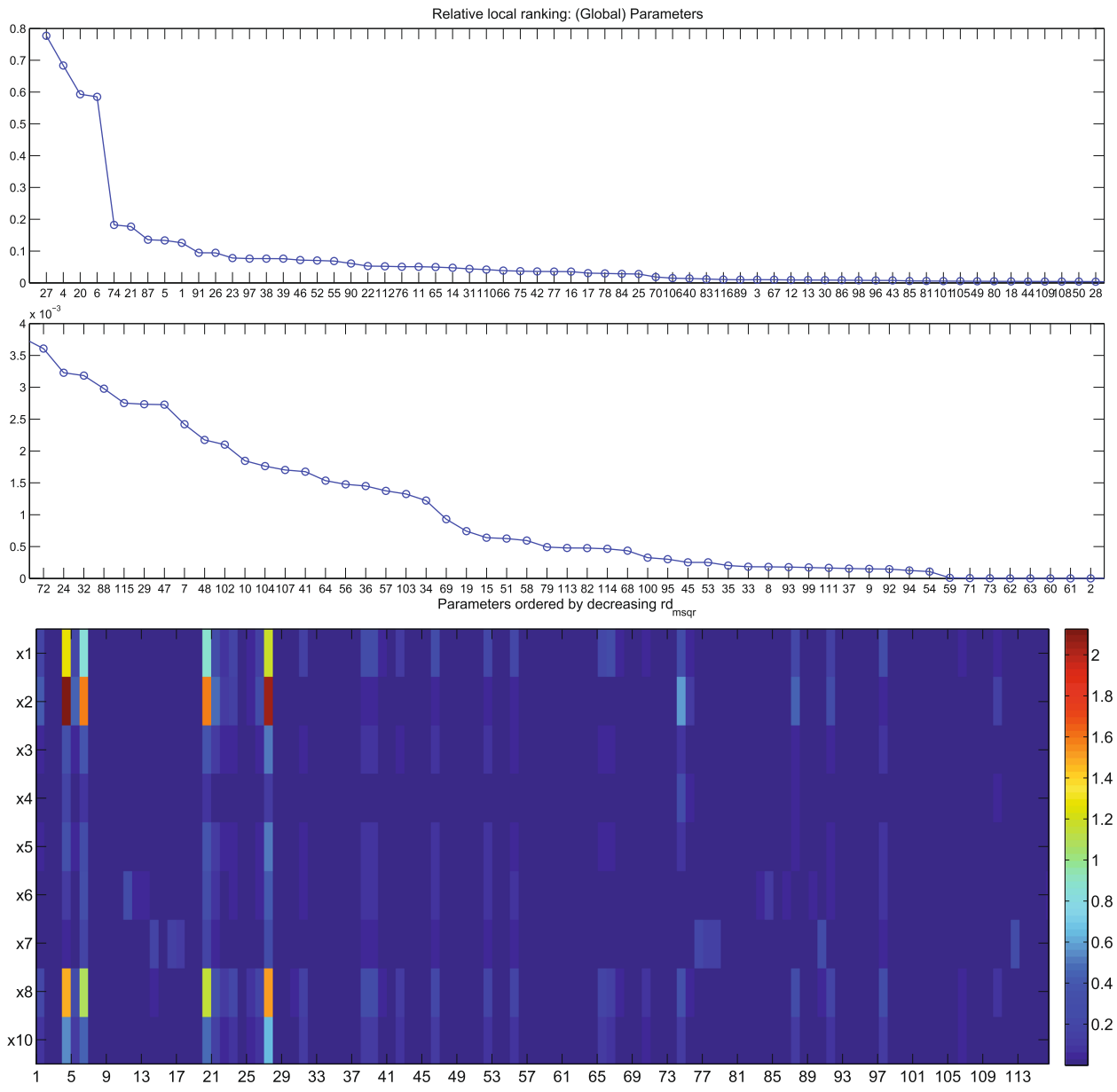

Figure 1 Benchmark 2: sensitivities. The two panels on top show the local rank of the parameters, i.e., the parameters ordered in decreasing order of their influence on the system's behaviour (δ_p^{msqr}, as defined in equations (9) and (10)). Note that the middle panel is a continuation of the upper one with a smaller y-axis scale. The array in the bottom panel shows the sensitivity of the 9 state variables (metabolite concentrations, in columns) of the model with respect to the 116 parameters. The colour bar in the right shows the sensitivity range: high sensitivities are plotted in red, low sensitivities in blue.

found) obtained with the DHC local method for benchmark B3. Similar outcomes were obtained with the other benchmarks and methods. Complete results for all the benchmarks and with different methods are included in the Additional file 1. In all cases, the number of local searches was fixed so that their overall CPU time was comparable to that consumed in optimizations where the global method eSS was used. While there was great variability in the results obtained for the different benchmarks, a conclusion was common to all of them: in all cases, the local methods were outperformed by the global optimization method eSS.

The convergence curves of the six benchmarks are shown in Figure 3. Results were obtained with the eSS method on a computer with Intel Xeon Quadcore processor, 2.50 GHz. It can be clearly noticed that, due to the

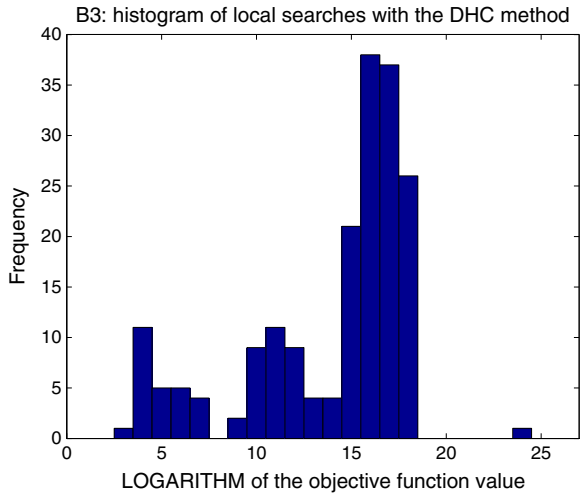

Figure 2 Benchmark 3. Histograms of local searches. The X axis shows the values of the solutions found by the DHC local method, and the Y axis shows their frequency. Of the total of 1000 local searches launched, only the 188 that converged are shown.

differences in size and complexity, the computational cost of estimating the parameters varies among benchmarks. Results show that they can be naturally classified in three different levels:

- B1 and B3 are the most expensive: in our computers, obtaining a reasonably good fit took at least one week.
- B5 and B6 are intermediate in terms of cost; a good fit could be obtained in one day.
- B2 and B4 are the least expensive, with good fits obtained in one or a few hours.

These computation times can be used as a reference to select the appropriate benchmarks to test a particular optimization method, depending on its focus and the available time. Due to the stochastic nature of the eSS algorithm, results may vary among optimization runs. Figure 4 shows the dispersion of 20 different optimization results for benchmark B4.

Table 2 summarizes the settings and outcomes of the parameter estimations with eSS, including the local method used for each problem. Note that, while DN2FB is generally recommended [65], we have realized that for large-scale problems it may not be the most efficient local method, due to the large number of evaluations needed to calculate the derivatives. Hence, for the problems considered here it is outperformed by other methods like DHC, SOLNP, or FMINCON.

One of the outcomes reported in Table 2 is the cumulative normalized root-mean-square error, $\sum$NRMSE. The root-mean-square error is a standard measure of the goodness of fit obtained for an observable which is defined as

$$\mathrm{RMSE}^o = \sqrt{\frac{\sum_{\epsilon=1}^{n_\epsilon} \sum_{s=1}^{n_s^{\epsilon,o}} \left(ym_s^{\epsilon,o} - y_s^{\epsilon,o}(p)\right)^2}{n_\epsilon \cdot n_s^{\epsilon,o}}} \tag{11}$$

with the same notation as in equation (1). To account for the different magnitudes of the observables it is useful to report the normalized root-mean-square error, NRMSE^o, which scales the RMSE^o by dividing it by the range of values of the observable:

$$\mathrm{NRMSE}^o = \frac{\mathrm{RMSE}^o}{max(ym^{\epsilon,o}) - min(ym^{\epsilon,o})} \tag{12}$$

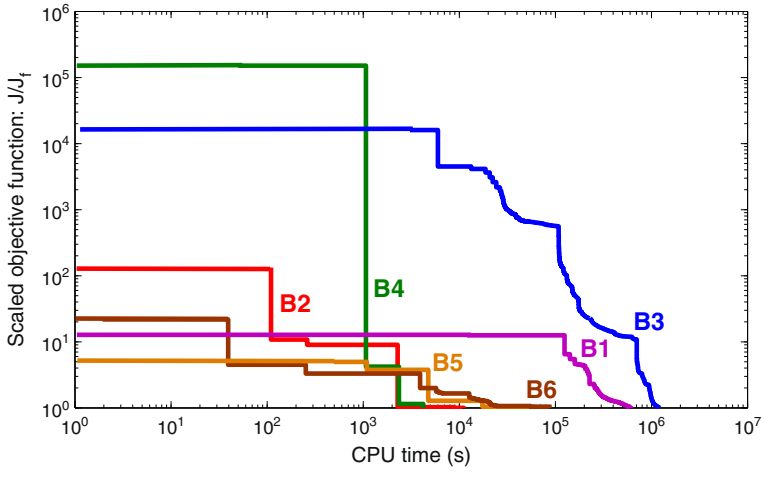

Figure 3 Convergence curves. Representative results of parameter estimation runs of the six benchmarks, carried out with the eSS method. The curves plot the (logarithmic) objective function value as a function of the (logarithmic) computation time. For ease of visualization, the values in the curves have been divided by the final value reached by each of them, i.e. the y axis plots J/J_f. Note that, since the benchmarks have different number of variables and data points, and different noise levels, the objective function values are not equivalent for different models. Results obtained on a computer with Intel Xeon Quadcore processor, 2.50 GHz, using Matlab 7.9.0.529 (R2009b) 32-bit.

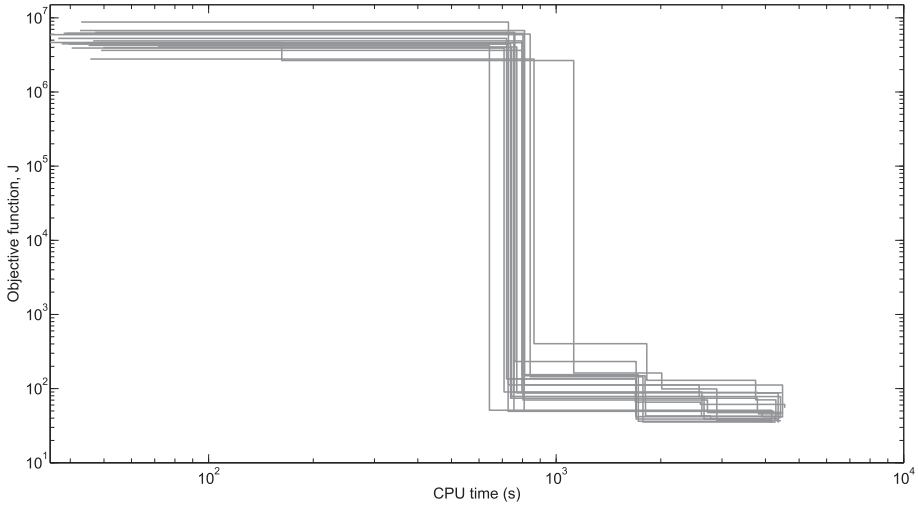

Figure 4 Dispersion of convergence curves. Results of 20 parameter estimation runs of the B4 benchmark (CHO cells) with the eSS method. The figures plot the objective function value as a function of the computation time (in log-log scale). Results obtained on a computer with Intel Xeon Quadcore processor, 2.50 GHz, using Matlab 7.9.0.529 (R2009b) 32-bit.

The cumulative normalized root-mean-square error, $\sum$NRMSE, is simply the sum of the NRMSE^o for all observables.

Note that, due to the realistic nature of most of these problems, there may be lack of identifiability and optimization may result in overfitting: that is, an optimal solution may be found that gives a better fit to the pseudoexperimental data than the one obtained with the nominal parameter vector used to generate the data. This is explained because, in the presence of measurement noise, the optimal solution manages to fit partially not only the system dynamics, but also the noise itself–which of course cannot be achieved by the nominal solution. Hence in the results reported in Table 2 the optimal objective function value (J_f) is sometimes smaller (i.e. better) than the nominal one (J_{nom}). This may also happen

Table 2 Parameter estimation with eSS (AMIGO implementation): settings and results

Model ID	B1	B2	B3	B4	B5	B6
p^U	$5 \cdot p_{nom}$	$10 \cdot p_{nom}^{(ex)}$	$10 \cdot p_{nom}^{(ex)}$	$5 \cdot p_{nom}$	varying	varying
p^L	$0.2 \cdot p_{nom}$	$0.1 \cdot p_{nom}^{(ex)}$	$0.1 \cdot p_{nom}^{(ex)}$	$0.2 \cdot p_{nom}$	varying	varying
Local method	DHC	FMINCON	none	FMINCON	DHC	FMINCON
CPU time	≈170 hours	≈3 hours	≈336 hours	≈1 hour	≈16 hours	≈24 hours
Evaluations	$6.9678 \cdot 10^5$	$9.0728 \cdot 10^4$	$7.2193 \cdot 10^6$	$1.6193 \cdot 10^5$	$8.8393 \cdot 10^4$	$2.0751 \cdot 10^6$
J_0	$5.8819 \cdot 10^9$	$3.1136 \cdot 10^4$	$4.6930 \cdot 10^{16}$	$6.6034 \cdot 10^8$	$3.1485 \cdot 10^4$	$8.5769 \cdot 10^5$
J_f	$1.3753 \cdot 10^4$	$2.3390 \cdot 10^2$	$3.7029 \cdot 10^{-1}$	$4.5718 \cdot 10^1$	$3.0725 \cdot 10^3$	$1.0833 \cdot 10^5$
J_{nom}	$1.0846 \cdot 10^6$	–	0	$3.9068 \cdot 10^1$	$4.2737 \cdot 10^3$	–
$\sum$NRMSE$_0$	$3.5834 \cdot 10^1$	$8.5995 \cdot 10^{-2}$	$3.5457 \cdot 10^1$	$4.8005 \cdot 10^1$	$4.0434 \cdot 10^1$	$2.3808 \cdot 10^2$
$\sum$NRMSE$_f$	5.7558	2.4921	$2.9298 \cdot 10^{-1}$	2.8010	$2.7430 \cdot 10^1$	$1.6212 \cdot 10^2$
$\sum$NRMSE$_{nom}$	3.8203	–	0	2.8273	$3.0114 \cdot 10^1$	–

Optimization settings and results obtained for each of the benchmarks with the eSS method, using the implementation provided in the AMIGO toolbox. In some cases the lower (p^L) and upper (p^U) bounds in the parameters are specified as a function of the nominal parameter vector, p_{nom}. There may be exceptions to these bounds, in cases where it makes sense biologically to have a different range of values (e.g. Hill coefficients in the range of 1–12). Cases with exceptions are marked by $^{(ex)}$. In other cases all the parameters have specific bounds; this is marked as "varying". The initial objective function value, J_0, corresponds to the parameter vector p_0 used as initial guess in the optimizations, which is randomly selected between the bounds p^L and p^U. The only exception is benchmark B2, where p_0 is the parameter vector reported in the original publication. The final value achieved in the optimizations is J_f, and the value obtained with the nominal parameter vector is J_{nom}. More details about the definition of the objective functions J are given in section "Problem statement". $\sum$NRMSE is the cumulative normalized root-mean-square error as defined in eq. (12); the subscripts ($_{0,\ f,\ nom}$) have the same meaning as in the objective functions J. Results obtained on a computer with Intel Xeon Quadcore processor, 2.50 GHz, using Matlab 7.9.0.529 (R2009b) 32-bit.

with the $\sum$NRMSE values. Note however that, since the objective functions used in the calibration (J) and the $\sum$NRMSE are different metrics, their behavior may be different. For example, for B1 $J_f < J_{nom}$ and $\sum \text{NRMSE}_f > \sum \text{NRMSE}_{nom}$, while for B4 the opposite is true: $J_f > J_{nom}$ and $\sum \text{NRMSE}_f < \sum \text{NRMSE}_{nom}$.

As an example of the fit between data and model output that is obtained after calibration, let us consider benchmark B5, which uses pseudoexperimental data corresponding to ten different experiments. Figure 5 reports a good match between data and model output; notably, the algorithm manages to reproduce the oscillations in NFκB.

The fit can also be represented with histograms of the residuals, which show the distribution of the errors in the state variables. This kind of plot can also be used for showing the errors in the recovered parameters when compared to the nominal (true) values. An alternative way of visualizing this relation is by plotting the predicted states (or parameter) values as a function of the true values. This results in a diagonal–like plot; the larger the deviations from the diagonal, the larger the prediction errors. When there are identifiability issues, the fit is typically better for the states than for the parameters, because a good fit to the data does not necessarily ensure that the correct parameters have been recovered. Examples of these plots are shown in Figure 6, which shows the fits obtained for benchmark B4.

Conclusions

To address the current lack of ready-to-run benchmarks for large-scale dynamic models in systems biology, we have presented here a collection of six parameter estimation problems. They cover the most common types, including metabolism, transcription, signal transduction, and development. The benchmarks are made available in a number of formats. As a common denominator, all of the models have been implemented in Matlab and C. When possible (i.e. for benchmarks B1–B5), model descriptions are also given in SBML. Ready-to-run implementations of all the benchmarks are provided in Matlab format (both with and without the AMIGO toolbox) and in COPASI (for benchmarks B1–B4). With these files it is straightforward to reproduce the results reported here.

More importantly, the benchmark files can be easily adapted to test new parameter estimation methods for which a Matlab, C, or COPASI implementation is available. The performance of an existing or newly developed

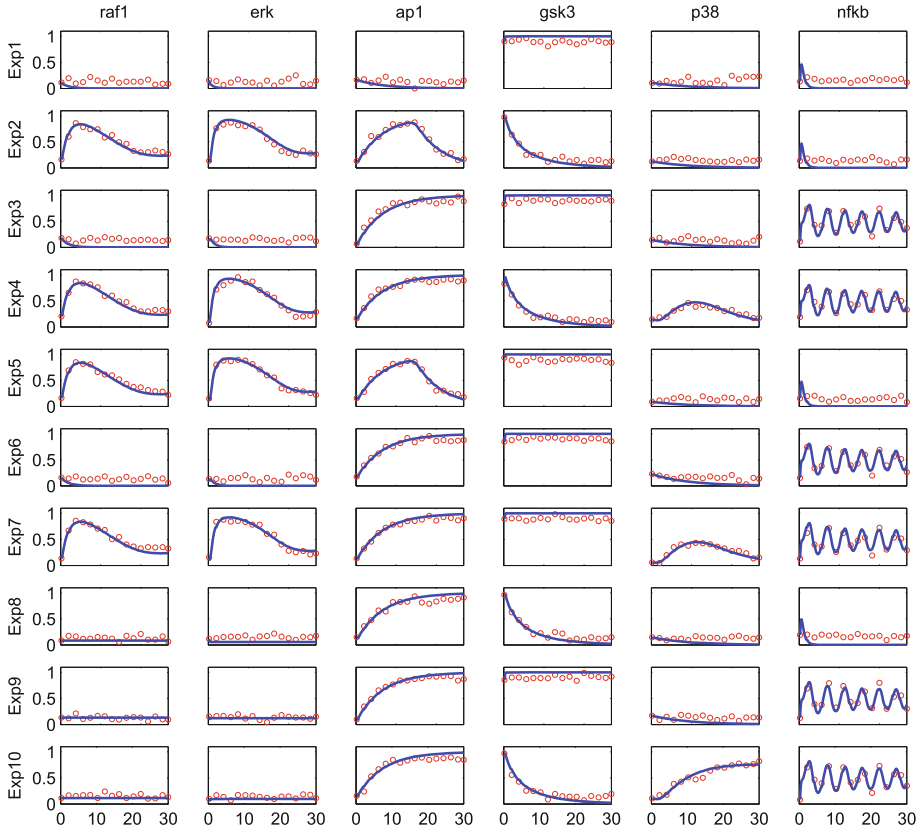

Figure 5 Benchmark 5. Data fits: time courses. Pseudo-experimental data (red circles) vs. optimal solution (solid blue lines) for the 6 observed states. X axis: time [minutes]. Y axis: activation level [0÷1].

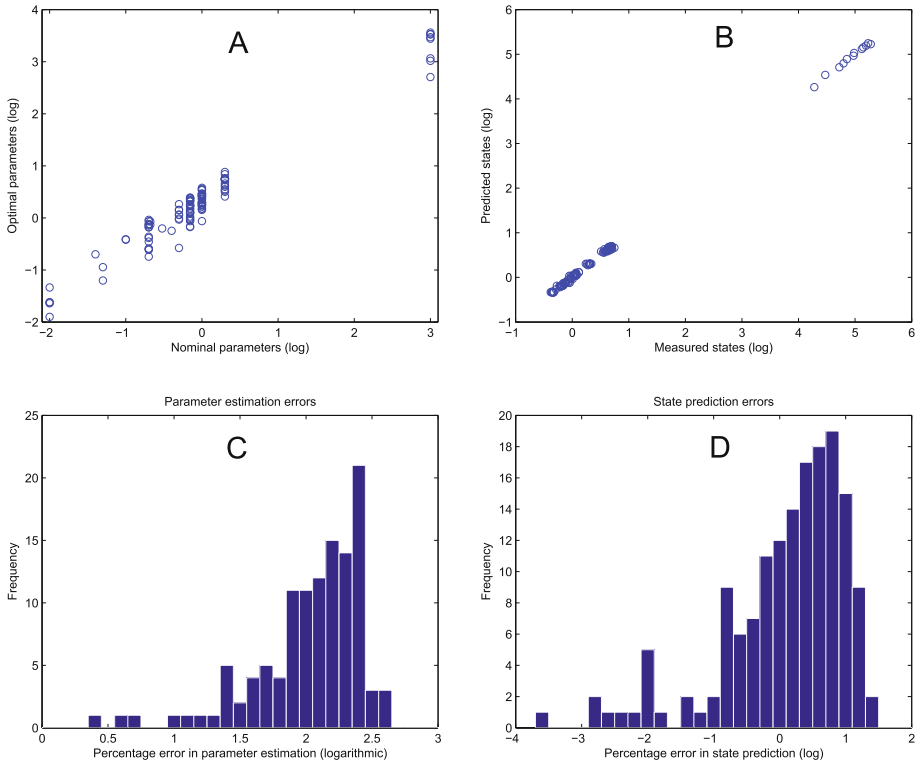

Figure 6 Benchmark 4, typical parameter estimation results. (A) Optimal vs. nominal parameters. **(B)** Pseudo-experimental ("measured states") vs. simulated data ("predicted states"). **(C)** Errors in the parameters: histogram of the differences between the nominal parameter vector and the optimal solution, in %. **(D)** Errors in the predictions: histogram of the difference between pseudo-experimental and simulated data, in %.

method can be evaluated by comparing its results with those reported here, as well as with those obtained by other methods. To this end, we have provided guidelines for comparing the performance of different optimizers. The problems defined here may also be used for educational purposes, running them as examples in classes or using them as assignments.

Finally, it should be noted that the utility of this collection goes beyond parameter estimation: the models provided here can also be used for benchmarking methods for optimal experimental design, identifiability analysis, sensitivity analysis, model reduction, and in the case of metabolic models also for metabolic engineering purposes.

Availability and requirements

Project name: BioPreDyn-Bench
Project home page: https://sites.google.com/site/biopredynbenchmarks/
Operating systems: Windows, Linux, Mac OSX
Programming languages: C, Matlab
Other requirements:

- For C implementations: GCC compiler, Matlab Compiler Runtime (the MCR is a free component which does not require a Matlab installation)
- For Matlab and AMIGO implementations: Matlab R2008 or newer
- For COPASI implementations: COPASI version 4.13 (Build 87) or newer
- For SBML: any software package capable of reading SBML

Please refer to the Additional file 1 for more details on these requirements and installation instructions

License: Artistic License 2.0

Restrictions for non-academic use: None

Additional files

Additional file 1: Describes the benchmark models in further detail and reports additional results.

Additional file 2: The first part of a compressed folder with the benchmark implementations.

Additional file 3: The second part of a compressed folder with the benchmark implementations.

Additional file 4: The first part of a compressed folder with the AMIGO toolbox.

Additional file 5: The second part of a compressed folder with the AMIGO toolbox.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
JRB, PM, and JJ conceived of the study. JRB and AFV coordinated the study. KS, SB, JS, DCS, AC, JSR, KM, PM, and JJ contributed with models. AFV, DH, KS, and DCS formatted the models and tested the implementations. AFV and DH carried out the computations. AFV, EBC, and JRB analyzed the results. AFV and JRB drafted the manuscript. KS, SB, AC, JSR, KM, EBC, and JJ helped to draft the manuscript. All authors read and approved the final manuscript.

Acknowledgments
This work was supported by the EU project "BioPreDyn" (EC FP7-KBBE-2011-5, grant number 289434). We acknowledge support of the publication fee by the CSIC Open Access Publication Support Initiative through its Unit of Information Resources for Research (URICI). We would like to thank Attila Gábor for reading the manuscript, providing critical comments, and finding bugs in the codes; David Rodríguez Penas for helping in debugging the codes; and Thomas Cokelaer for providing the SBML qual file of model B5.

Author details
[1]Bioprocess Engineering Group, IIM-CSIC, Eduardo Cabello 6, 36208 Vigo, Spain. [2]European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Trust Genome Campus, CB10 1SD Hinxton, Cambridge, UK. [3]School of Computer Science, Manchester Centre for Integrative Systems Biology, The University of Manchester, 131 Princess Street, M1 7DN Manchester, UK. [4]Insilico Biotechnology AG, Meitnerstraße 8, 70563 Stuttgart, Germany. [5]EMBL/CRG Research Unit in Systems Biology, Centre for Genomic Regulation (CRG), Dr. Aiguader 88, 08003 Barcelona, Spain. [6]Universitat Pompeu Fabra (UPF), Plaça de la Mercé, 10, 08002 Barcelona, Spain.

Received: 21 July 2014 Accepted: 15 January 2015
Published online: 20 February 2015

References

1. Link H, Christodoulou D, Sauer U. Advancing metabolic models with kinetic information. Curr Opin Biotechnol. 2014;29:8–14.
2. Almquist J, Cvijovic M, Hatzimanikatis V, Nielsen J, Jirstrand M. Kinetic models in industrial biotechnology–improving cell factory performance. Metab Eng. 2014;24:38–60.
3. Song H-S, DeVilbiss F, Ramkrishna D. Modeling metabolic systems: the need for dynamics. Curr Opin Chem Eng. 2013;2(4):373–82.
4. Jaeger J, Monk N. Bioattractors: Dynamical systems theory and the evolution of regulatory processes. J Physiol (Lond). 2014;592:2267–81.
5. Villaverde AF, Banga JR. Reverse engineering and identification in systems biology: strategies, perspectives and challenges. J R Soc Interface. 2014;11(91):20130505.
6. van Riel N.A.W. Dynamic modelling and analysis of biochemical networks: mechanism-based models and model-based experiments. Brief Bioinform. 2006;7(4):364–74.
7. Jaqaman K, Danuser G. Linking data to models: data regression. Nat Rev Mol Cell Biol. 2006;7(11):813–19.
8. Banga J, Balsa-Canto E. Parameter estimation and optimal experimental design. Essays Biochem. 2008;45:195.
9. Ashyraliyev M, Fomekong-Nanfack Y, Kaandorp JA, Blom JG. Systems biology: parameter estimation for biochemical models. FEBS J. 2008;276(4):886–902.
10. Vanlier J, Tiemann C, Hilbers P, van Riel N. Parameter uncertainty in biochemical models described by ordinary differential equations. Math Biosci. 2013;246(2):305–14.
11. Moles CG, Mendes P, Banga JR. Parameter estimation in biochemical pathways: a comparison of global optimization methods. Genome Res. 2003;13(11):2467.
12. Banga JR. Optimization in computational systems biology. BMC Syst Biol. 2008;2:47.
13. Mendes P, Sha W, Ye K. Artificial gene networks for objective comparison of analysis algorithms. Bioinformatics. 2003;19(suppl 2):122–29.
14. Kremling A, Fischer S, Gadkar K, Doyle FJ, Sauter T, Bullinger E, et al. A benchmark for methods in reverse engineering and model discrimination: problem formulation and solutions. Genome Res. 2004;14(9):1773–85.
15. Camacho D, Vera Licona P, Mendes P, Laubenbacher R. Comparison of reverse-engineering methods using an in silico network. Ann N Y Acad Sci. 2007;1115(1):73–89.
16. Gennemark P, Wedelin D. Benchmarks for identification of ordinary differential equations from time series data. Bioinformatics. 2009;25(6):780–86.
17. Haynes BC, Brent MR. Benchmarking regulatory network reconstruction with grendel. Bioinformatics. 2009;25(6):801–07.
18. Marbach D, Schaffter T, Mattiussi C, Floreano D. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. J Comput Biol. 2009;16(2):229–39.
19. Schaffter T, Marbach D, Floreano D. Genenetweaver: in silico benchmark generation and performance profiling of network inference methods. Bioinformatics. 2011;27(16):2263–70.
20. Meyer P, Cokelaer T, Chandran D, Kim KH, Loh P-R, Tucker G, et al. Network topology and parameter estimation: from experimental design methods to gene regulatory network kinetics using a community based approach. BMC Syst Biol. 2014;8(1):13.
21. Auger A, Hansen N, Schoenauer M. Benchmarking of continuous black box optimization algorithms. Evol Comput. 2012;20(4):481.
22. Dolan ED, Moré JJ, Munson TS. Benchmarking optimization software with cops 3.0. Argonne National Laboratory Technical Report ANL/MCS-TM-273, 9700 South Cass Avenue, Argonne, Illinois 60439, USA. 2004.
23. Chassagnole C, Noisommit-Rizzi N, Schmid JW, Mauch K, Reuss M. Dynamic modeling of the central carbon metabolism of escherichia coli. Biotechnol Bioeng. 2002;79(1):53–73.
24. Kotte O, Zaugg JB, Heinemann M. Bacterial adaptation through distributed sensing of metabolic fluxes. Mol Syst Biol. 2010;6(1):355.
25. Smallbone K, Mendes P. Large-scale metabolic models: From reconstruction to differential equations. Ind Biotech. 2013;9(4):179–84.
26. Villaverde AF, Bongard S, Schmid J, Müller D, Mauch K, Balsa-Canto E, et al. High-confidence predictions in systems biology dynamic models. In: Advances in Intelligent and Soft-Computing, vol. 294. Switzerland: Springer; 2014. p. 161–71.
27. MacNamara A, Terfve C, Henriques D, Bernabé BP, Saez-Rodriguez J. State–time spectrum of signal transduction logic models. Phys Biol. 2012;9(4):045003.
28. Jaeger J, Surkova S, Blagov M, Janssens H, Kosman D, Kozlov KN, et al. Dynamic control of positional information in the early drosophila embryo. Nature. 2004;430(6997):368–71.
29. Crombach A, Wotton KR, Cicin-Sain D, Ashyraliyev M, Jaeger J. Efficient reverse-engineering of a developmental gene regulatory network. PLoS Comput Biol. 2012;8(7):1002589.
30. Ashyraliyev M, Siggens K, Janssens H, Blom J, Akam M, Jaeger J. Gene circuit analysis of the terminal gap gene huckebein. PLoS Comput Biol. 2009;5(10):1000548.
31. Krause F, Schulz M, Swainston N, Liebermeister W. Sustainable model building the role of standards and biological semantics. Methods Enzymol. 2011;500:371–95.
32. Hucka M, Finney A, Sauro H. M, Bolouri H, Doyle JC, Kitano H, et al. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. Bioinformatics. 2003;19(4):524–31.
33. Balsa-Canto E, Banga JR. Amigo, a toolbox for advanced model identification in systems biology using global optimization. Bioinformatics. 2011;27(16):2311–3.
34. Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, et al. Copasi – a complex pathway simulator. Bioinformatics. 2006;22(24):3067–74.
35. Balsa-Canto E, Alonso A, Banga JR. An iterative identification procedure for dynamic modeling of biochemical networks. BMC Syst Biol. 2010;4:11.
36. Egea JA, Martí R, Banga JR. An evolutionary method for complex-process optimization. Comput Oper Res. 2010;37(2):315–24.
37. Villaverde A, Egea J, Banga J. A cooperative strategy for parameter estimation in large scale systems biology models. BMC Syst Biol. 2012;6(1):75.
38. Egea J, Henriques D, Cokelaer T, Villaverde A, MacNamara A, Danciu D.-P, et al. Meigo: an open-source software suite based on metaheuristics for global optimization in systems biology and bioinformatics. BMC Bioinformatics. 2014;15:136.
39. Dolan E. D, Moré J. J. Benchmarking optimization software with performance profiles. Math Program, Ser A. 2002;91(2):201–13.

40. Walter E, Pronzato L. Identification of parametric models from experimental data. Communications and control engineering series. London, UK: Springer; 1997.
41. Gadkar KG, Gunawan R, Doyle FJ. Iterative approach to model identification of biological networks. BMC Bioinformatics. 2005;6(1):155.
42. Chiş O-T, Banga JR, Balsa-Canto E. Structural identifiability of systems biology models: a critical comparison of methods. PLoS ONE. 2011;6(11): 27755.
43. Chiş O, Banga J.R, Balsa-Canto E. Genssi: a software toolbox for structural identifiability analysis of biological models. Bioinformatics. 2011;27(18): 2610–1.
44. Becker K, Balsa-Canto E, Cicin-Sain D, Hoermann A, Janssens H, Banga JR, et al. Reverse-engineering post-transcriptional regulation of gap genes in drosophila melanogaster. PLoS Comput Biol. 2013;9(10):1003281.
45. Zak DE, Gonye GE, Schwaber JS, Doyle FJ. Importance of input perturbations and stochastic gene expression in the reverse engineering of genetic regulatory networks: insights from an identifiability analysis of an in silico network. Genome Res. 2003;13(11):2396–405.
46. Yue H, Brown M, Knowles J, Wang H, Broomhead DS, Kell DB. Insights into the behaviour of systems biology models from dynamic sensitivity and identifiability analysis: a case study of an nf-κb signalling pathway. Mol Biosyst. 2006;2(12):640–9.
47. Anguelova M, Cedersund G, Johansson M, Franzen C, Wennberg B. Conservation laws and unidentifiability of rate expressions in biochemical models. IET Syst Biol. 2007;1(4):230–7.
48. Srinath S, Gunawan R. Parameter identifiability of power-law biochemical system models. J Biotechnol. 2010;149(3):132–40.
49. Szederkényi G, Banga JR, Alonso AA. Inference of complex biological networks: distinguishability issues and optimization-based solutions. BMC Syst Biol. 2011;5(1):177.
50. Miao H, Xia X, Perelson AS, Wu H. On identifiability of nonlinear ode models and applications in viral dynamics. SIAM Rev. 2011;53(1):3–39.
51. Jia G, Stephanopoulos G, Gunawan R. Incremental parameter estimation of kinetic metabolic network models. BMC Syst Biol. 2012;6(1):142.
52. Cedersund G. Conclusions via unique predictions obtained despite unidentifiability–new definitions and a general method. FEBS J. 2012;279(18):3513–27.
53. Berthoumieux S, Brilli M, Kahn D, De Jong H, Cinquemani E. On the identifiability of metabolic network models. J Math Biol. 2013;67(6-7): 1795–832.
54. DiStefano III J. Dynamic systems biology modeling and simulation. Waltham, MA, USA: Academic Press; 2014.
55. Sontag ED. For differential equations with r parameters, 2r+ 1 experiments are enough for identification. J Nonlinear Sci. 2002;12(6):553–83.
56. Heavner BD, Smallbone K, Barker B, Mendes P, Walker LP. Yeast 5–an expanded reconstruction of the saccharomyces cerevisiae metabolic network. BMC Syst Biol. 2012;6(1):55.
57. Smallbone K, Simeonidis E. Flux balance analysis: A geometric perspective. J Theor Biol. 2009;258(2):311–5.
58. Liebermeister W, Uhlendorf J, Klipp E. Modular rate laws for enzymatic reactions: thermodynamics, elasticities, and implementation. Bioinformatics. 2010;26(12):1528–34.
59. Li C, Donizelli M, Rodriguez N, Dharuri H, Endler L, Chelliah V, et al. BioModels Database: an enhanced, curated and annotated resource for published quantitative kinetic models. BMC Syst Biol. 2010;4:92.
60. Wurm FM. Production of recombinant protein therapeutics in cultivated mammalian cells. Nat Biotechnol. 2004;22(11):1393–98.
61. Wittmann DM, Krumsiek J, Saez-Rodriguez J, Lauffenburger DA, Klamt S, Theis FJ. Transforming boolean models to continuous models: methodology and application to t-cell receptor signaling. BMC Syst Biol. 2009;3(1):98.
62. Chaouiya C, Bérenguier D, Keating SM, Naldi A, Van Iersel M. P, Rodriguez N, et al. Sbml qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools. BMC Syst Biol. 2013;7(1):135.
63. Mjolsness E, Sharp DH, Reinitz J. A connectionist model of development. J Theor Biol. 1991;152(4):429–53.
64. Reinitz J, Sharp DH. Mechanism of eve stripe formation. Mech Dev. 1995;49(1):133–58.
65. Rodríguez-Fernández M, Egea JA, Banga JR. Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. BMC Bioinformatics. 2006;7:483.